The Todo.txt Format

leun4m edited this page on May 7 · 21 revisions

A complete primer on the whys and hows of `todo.txt` .

Why plain text?

Plain text is software and operating system agnostic. It's searchable, portable, lightweight and easily manipulated. It's unstructured. It works when someone else's web server is down or your Outlook .PST file is corrupt. There's no exporting and importing, no databases or tags or flags or stars or prioritizing or [Insert company name here]-induced rules on what you can and can't do with it.

The 3 axes of an effective todo list

Using special notation in todo.txt, you can create a list that's sliceable by 3 key axes.

Priority. Your todo list should be able to tell you what's the next most important thing for you to get done – either by project or by context or overall. You can optionally assign tasks a priority that'll bubble them up to the top of the list.

Project. The only way to move a big project forward is to tackle a small subtask associated with it. Your todo.txt should be able to list out all the tasks specific to a project.

In order to move along a project like "Cleaning out the garage," my task list should give me the next logical action to take in order to move that project along. "Clean out the garage" isn't a good todo item; but "Call Goodwill to schedule pickup" in the "Clean out garage" project is.

Context. *Getting Things Done* author David Allen suggests splitting up your task lists by context – ie, the place and situation where you'll work on the job. Messages that you need to send go in the "@email" context; calls to be made "@phone", household projects "@home."

That way, when you've got a few minutes in the car with your cell phone, you can easily check your "@phone" tasks and make a call or two while you have the opportunity.

This is all possible inside todo.txt.

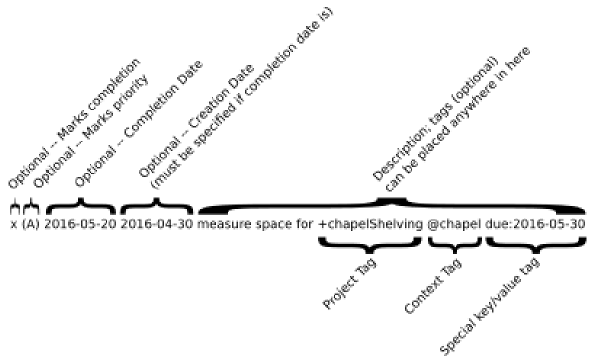
Todo.txt format rules

Your Todo.txt is a plain text file. To take advantage of structured task metadata like priority, projects, context, creation and completion date, there are a few simple but flexible file format rules.

Philosophically, the Todo.txt file format has two goals:

- The file contents should be human-readable without requiring any tools other than a plain text viewer or editor.
- A user can manipulate the file contents in a plain text editor in sensible, expected ways. For example, a text editor that can sort lines alphabetically should be able to sort your task list in a meaningful way.

(These two goals are why, for example, lines start with priority and/or dates, so that they are easily sorted by priority or time, and completed items are marked with an x, which both sorts at the bottom of an alphabetical list AND looks like a filled-in checkbox.)


Optional -- Marks completion
Optional -- Marks priority
Optional -- Completion Date
Optional -- Creation Date (must be specified if completion date is)
Description; tags (optional) can be placed anywhere in here
x (A) 2016-05-20 2016-04-30 measure space for +chapelShelving @chapel due:2016-05-30
Project Tag
Context Tag
Special key/value tag

The first and most important rule of todo.txt is: ***A single line in your todo.txt text file represents a single task.***

Here are the rest.

Incomplete Tasks: 3 Format Rules

The beauty of todo.txt is that it’s completely unstructured; the fields you can attach to each task are only limited by your imagination. To get started, use special notation to indicate task context (like `@phone`), project (like `+GarageSale`) and priority (like `(A)`). So, a todo.txt file might look like this:

```
(A) Thank Mom for the meatballs @phone
(B) Schedule Goodwill pickup +GarageSale @phone
Post signs around the neighborhood +GarageSale
@GroceryStore Eskimo pies
```

A script that perhaps slices out the `@phone` contextual items and emails them to your mobile phone, for instance, would just output:

```
(A) Thank Mom for the meatballs @phone
(B) Schedule Goodwill pickup +GarageSale @phone
```

A call to `todo.sh` to just see the garage sale project items would return:

```
(B) Schedule Goodwill pickup +GarageSale @phone
Post signs around the neighborhood +GarageSale
```

There are three formatting rules for current to-do’s.

Rule 1: If priority exists, it ALWAYS appears first.

The priority is an uppercase character from A-Z enclosed in parentheses and followed by a space.

For example, this is a task with an A priority:

`(A) Call Mom`

These tasks have no priority:

```
Really gotta call Mom (A) @phone @someday
(b) Get back to the boss
(B)->Submit TPS report
```

Rule 2: A task’s creation date may optionally appear directly after priority and a space.

If there is no priority, the creation date appears first. If the creation date exists, it should be in the format YYYY-MM-DD.

These tasks have creation dates:

```
2011-03-02 Document +TodoTxt task format
(A) 2011-03-02 Call Mom
```

This task doesn’t have a creation date:

```
(A) Call Mom 2011-03-02
```

Rule 3: Contexts and Projects may appear anywhere in the line *after* priority/prepended date.

A context is preceded by a single space and an @ sign. A project is preceded by a single space and a plus + sign. A project or context contains any non-whitespace character. A task may have zero, one, or more than one projects and contexts included in it.

For example, this task is part of the +Family and +PeaceLoveAndHappiness projects as well as the @iphone and @phone contexts:

```
(A) Call Mom +Family +PeaceLoveAndHappiness @iphone @phone
```

This task has no contexts in it:

```
Email SoAndSo at soandso@example.com
```

This task has no projects in it:

```
Learn how to add 2+2
```

Complete Tasks: 2 Format Rules

Two things indicate that a task has been completed.

Rule 1: A completed task starts with an x.

If a task starts with an x (case-sensitive lowercase) followed directly by a space, it is complete.

This is a complete task:

```
x 2011-03-03 Call Mom
```

These are not complete tasks.

```
xylophone lesson
X 2012-01-01 Make resolutions
(A) x Find ticket prices
```

We use a lowercase x so that completed tasks sort to the bottom of the task list using standard sort tools.

Rule 2: The date of completion appears directly after the x, separated by a space.

For example:

```
x 2011-03-02 2011-03-01 Review Tim's pull request +TodoTxtTouch @github
```

If you've prepended the creation date to your task, on completion it will appear directly after the completion date. This is so your completed tasks sort by date using standard sort tools. Many Todo.txt clients discard priority on task completion. To preserve it, use the key:value format described below (for example, pri:A)

With the completed date (required), if you've used the prepended date (optional), you can calculate how many days it took to complete a task.

Add-on File Format Definitions

The Todo.txt CLI is extensible with add-ons. An add-on may define its own additional formatting rules for extra metadata. In general, add-on developers should use the format `key:value` to define additional metadata, for example, `due:2010-01-02` as a due date. Both key and value must consist of non-whitespace characters, which are not colons. Only one colon separates the key and value.

Other notations

With `todo.sh` , you can choose any unique keyword, and search by it. For example, to indicate you're waiting on something to complete a task, append the word `WAIT` to the item in `todo.txt` . Others like to add due dates to a task, `DUE:2006-08-01` . It's completely up to you.

Handy Tip: To view items by keyword in the Todo.txt Command Line interface, do `todo.sh list yourkeyword` .